Программирование

Python

Работа с текстом

Кафедра ИВТ и ПМ
ЗабГУ

2018

План

Регулярные выражения. Введение.

XML и HTML

docx

Обработка текстов на естественном языке

Ссылки и литература

Outline

Регулярные выражения

Регулярные выражения

Регулярные выражения (regular expressions) — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов (wildcard characters).

Регулярные выражения

Для поиска используется строка-образец (**pattern**, «шаблоном», «маской»), состоящая из символов и метасимволов и задающая правило поиска.

Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

Регулярные выражения

Краткая информация о способах задания шаблона приведена в справке по модулю.

```
help( re )
```

Регулярные выражения

- $\backslash w$ - цифры, символы верхнего, нижнего регистра и "_"
- $\backslash s$ - пробел
- $\backslash d$ - число
- $\backslash$ - управляющий символ

Аналогично можно ограничивать поиск символами, которые НЕ должны встретится

- $\backslash W$ - не цифры, не символы верхнего, нижнего регистра и не "_"
- $\backslash S$ - не пробел
- $\backslash D$ - не число
- $\backslash C$ - не управляющий символ

Регулярные выражения

Если требуется задать собственный класс символов то можно использовать перечисления.

Перечисления задаются в квадратных скобках. Одно перечисление в фигурных скобках задаёт все возможные значения для *одного* символа.

Любая нечётная цифра

`[13579]`

Первые три буквы русского алфавита, либо заглавные либо строчные.

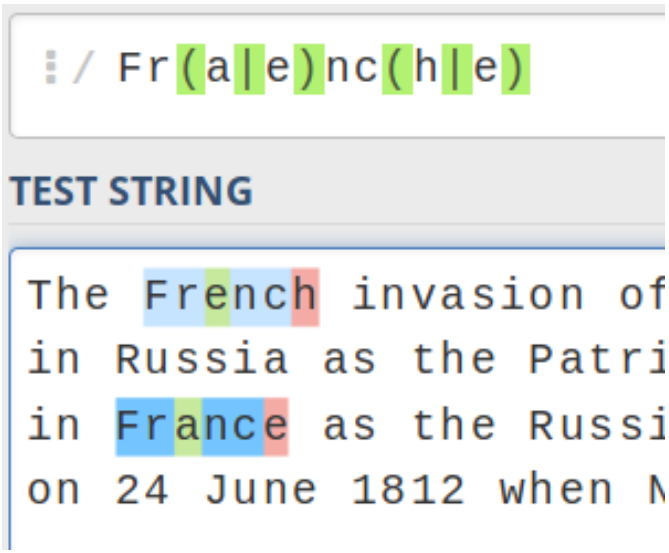
`[а-вА-В]`

Регулярные выражения

Если требуется найти найти похожие слова, отличающиеся одной или несколькими буквами можно задать шаблон таким образом:

```
Fr(a|e)nc(h|e)
```

Здесь запись (a|e) обозначает один символ, который может быть либо *a*, либо *e*



Регулярные выражения

Кванторы задают число повторений заданного символа или их группы

- Ровно три цифры
  `\d{3}`
- Слово длинной от 4 до 6 букв в нижнем регистре
  `[a-z]{4,6}`
- Слово в нижнем регистре из трёх букв или больше
  `[a-z]{3,}`
- 0 или 1 перечисленных символов
  `[qwerty]?`
- 0 или сколько угодно перечисленных символов
  `[qwerty]*`

Последние выражения не применяются отдельно, так как им может соответствовать пустая строка.

Регулярные выражения

```
import re

text = "The French invasion of Russia, known \
in Russia as the Patriotic War of 1812 and \
in France as the Russian Campaign, began \
on 24 June 1812 when Napoleon's Grande Armée \
crossed the Neman Rive"

res = re.search("\d\d\d\d", text)
if res:
    print("Найденная подстрока: " + res.group(0))
    print("Позиция подстроки: " + str(res.span()))
```

Найденная подстрока: 1812

Позиция подстроки: (71, 75)

Регулярные выражения в текстовых редакторах

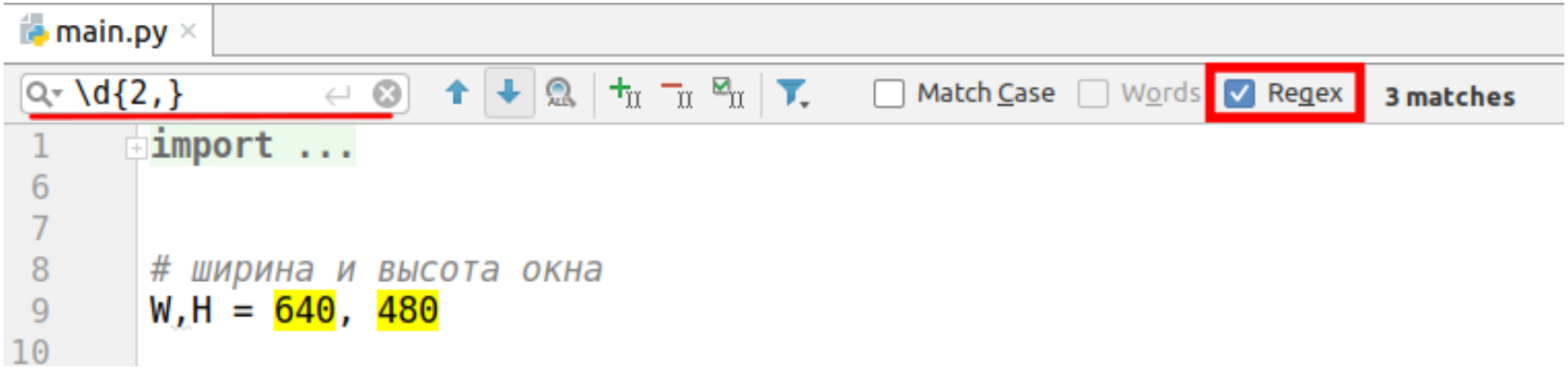
Регулярные выражения используются для гибкого поиска в больших текстах.

Например в программах Sublime text, atom и PyCharm можно включить поддержку регулярных выражений в обычном поиске.

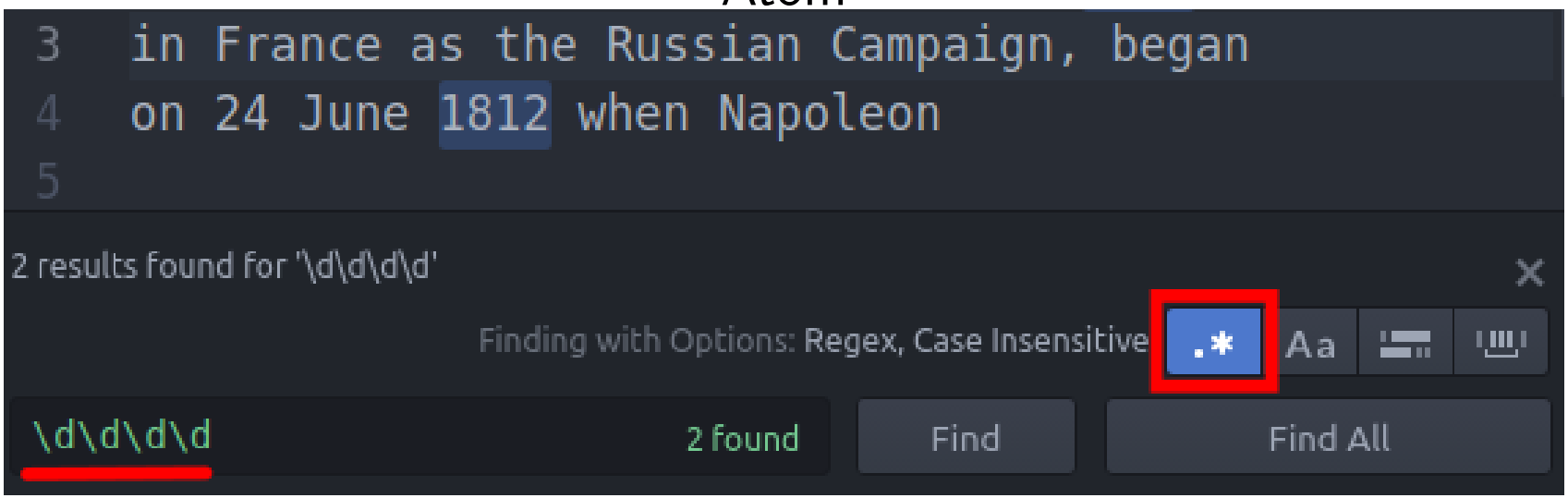
В Linux для поиска по регулярному выражению используется утилиты grep или egrep.

Регулярные выражения в текстовых редакторах

PyCharm



Atom



Примеры

```
8(\s|-)\d\d\d(\s|-)\d\d\d(\s|-)\d\d\d\d
```

Примеры

```
8(\s|-)\d\d\d(\s|-)\d\d\d(\s|-)\d\d\d\d
```

Поиск номера телефона вида 8-999-123-456. Вместо тире могут быть пробелы.

```
[А-Я]. [А-Я]. [А-Я][а-я]{2,}
```

Примеры

```
8(\s|-)\d\d\d(\s|-)\d\d\d(\s|-)\d\d\d\d
```

Поиск номера телефона вида 8-999-123-456. Вместо тире могут быть пробелы.

```
[А-Я]. [А-Я]. [А-Я][а-я]{2,}
```

Поиск строк вида "И. О. Фамилия"

Нечёткий поиск

```
import regex

# поиск подстроки amazing в строке amaging
# допустима одна или меньше опечатка ( e<=1 )
res = regex.match('(amazing){e<=1}', 'amaging')

if res:
    print("Найденная подстрока: " + res.group(0))
    print("Позиция подстроки: " + str(res.span()))
    print("Число опечаток, лишних вставок и недостающих символов: "
          + str(res.fuzzy_counts))
    print("Число позиций с опечатками, лишними вставками \
и недостающими символами: " + str(res.fuzzy_changes))
```

Найденная подстрока: amaging
Позиция подстроки: (0, 7)
Число опечаток, лишних вставок и недостающих символов: (1, 0, 0)
Число позиций с опечатками, лишними вставками и недостающими символами: ([3], [], [])

Применение

Помимо поиска подстрок в текстах регулярные выражения применяются для замены подстрок, а также для проверки введёных пользователем данных.

Например с помощью регулярного выражения можно проверить введёл ли корректный email-адрес, номер телефона или номер банковской карты.

Применение

Регулярные выражения учитывающие все аспекты искомой подстроки могут быть очень сложными и трудными в понимании. Поэтому следует применять и другие способы обработки текстовой информации, где регулярное выражение может использоваться только на одном их этапов.

Например можно ограничивать области текстов к которым применяются регулярные выражения если известна структура документа.
Если идёт речь о технических форматах документов, например html, то лучше использовать специальные методы разбора html документов.

Нечёткий поиск

Три типа ошибок

- “i” - вставлен лишний символ
- “d” - нужный символ пропущен (удалён)
- “s” - символ замещён

“e” обозначает любую из этих ошибок

Нечёткий поиск

Примеры:

i<=3 permit at most 3 insertions, but no other types
d<=3 permit at most 3 deletions, but no other types
s<=3 permit at most 3 substitutions, but no other types
i<=1,s<=2 permit at most 1 insertion and at most 2 substitutions, but no deletions
e<=3 permit at most 3 errors
1<=e<=3 permit at least 1 and at most 3 errors
i<=2,d<=2,e<=3 permit at most 2 insertions, at most 2 deletions, at most 3 errors in total, but no substitutions

pypi.org/project/regex - regex docs

Outline

Outline

Outline

Outline

Ссылки и литература

- regex101.com - online regex tester
- http://www.exlab.net/files/tools/sheets/regexp/regexp.pdf - шпаргалка по регулярным выражениям

Ссылки и литература

Ссылка на слайды

github.com/VetrovSV/Programming